04   EXPLOITING SOFTWARE DIVERSIFICATION FOR WEBASSEMBLY

■ 4.1   WASM-MUTATE WebAssembly malware evasion

TODO The malware evasion paper

■ 4.1.2 Objective

Test and evade the resilience of WebAssembly malware detectors mentioned in Subsection 2.1.6.

■ 4.1.3 Approach

TODO We use wasm-mutate   **TODO** How do we use it?   **TODO** Controlled and uncontrolled diversification.

■ 4.1.4 Results

■ 4.2   CROW: Automatic testing and fuzzing of WebAssembly consumers

TODO We explain the CVE. Make the explanation around "indirect memory diversification"

■ 4.2.2 Objective

Test compilers, interpreters, and runtimes for WebAssembly. In concrete wasmtime.

Make the history around the CVE.

---

[0] Comp. time 2023/10/02 13:03:43

■ 4.2.3 Approach

Use CROW to generate a set of Wasm binaries.

■ 4.2.4 Results

The CVE and the indirect memory diversification. This is a big insight.

■ 4.3 MEWE: Multivariant execution at the Edge

TODO Disturbing of execution time. Go around the web timing attacks. Attack model for MEWE.

■ 4.3.2 Threat model

- Software monoculture
    - Timing based attacks. Mentioned the Whitehat paper, mention the `https://arxiv.org/pdf/2210.10523.pdf` paper.

■ 4.3.3 Approach

- Use of MEWE

■ 4.3.4 Results

- Unpredictable variant given the execution time.

■ 4.4 WASM-MUTATE: Speculative Side-channel protection

TODO Go around the last paper

■ 4.4.2 Threat model

- Spectre timing cache attacks.
    - Rockiki paper on portable side channel in browsers.

■ 4.4.3 Approach

- Use of wasm-mutate

■ 4.4.4 Results

  - Diminshing of BER